Table 1 (cont')

```
/*
 *
 * print() -- only routine visible outside this module
 *
 * static:
 * getmat() -- trace back best path, count matches: print()
 * pr_align() -- print alignment of described in array p[]: print()
 * dumpblock() -- dump a block of lines with numbers, stars: pr_align()
 * nums() -- put out a number line: dumpblock()
 * putline() -- put out a line (name, [num], seq, [num]): dumpblock()
 * stars() - -put a line of stars: dumpblock()
 * stripname() -- strip any path and prefix from a seqname
 */

#include "nw.h"

#define SPC     3
#define P_LINE  256     /* maximum output line */
#define P_SPC   3       /* space between name or num and seq */

extern  _day[26][26];
int     olen;           /* set output line length */
FILE    *fx;            /* output file */

print()                                                          print
{
        int     lx, ly, firstgap, lastgap;      /* overlap */

        if ((fx = fopen(ofile, "w")) == 0) {
                fprintf(stderr,"%s: can't write %s\n", prog, ofile);
                cleanup(1);
        }
        fprintf(fx, "< first sequence: %s (length = %d)\n", namex[0], len0);
        fprintf(fx, "< second sequence: %s (length = %d)\n", namex[1], len1);
        olen = 60;
        lx = len0;
        ly = len1;
        firstgap = lastgap = 0;
        if (dmax < len1 - 1) {          /* leading gap in x */
                pp[0].spc = firstgap = len1 - dmax - 1;
                ly -= pp[0].spc;
        }
        else if (dmax > len1 - 1) {   /* leading gap in y */
                pp[1].spc = firstgap = dmax - (len1 - 1);
                lx -= pp[1].spc;
        }
        if (dmax0 < len0 - 1) {         /* trailing gap in x */
                lastgap = len0 - dmax0 -1;
                lx -= lastgap;
        }
        else if (dmax0 > len0 - 1) { /* trailing gap in y */
                lastgap = dmax0 - (len0 - 1);
                ly -= lastgap;
        }
        getmat(lx, ly, firstgap, lastgap);
        pr_align();
}
```

Table 1 (cont')

```
/*
 * trace back the best path, count matches
 */
static
getmat(lx, ly, firstgap, lastgap)                                    getmat
        int     lx, ly;                 /* "core" (minus endgaps) */
        int     firstgap, lastgap;      /* leading trailing overlap */
{
        int             nm, i0, i1, siz0, siz1;
        char            outx[32];
        double          pct;
        register        n0, n1;
        register char   *p0, *p1;

        /* get total matches, score
         */
        i0 = i1 = siz0 = siz1 = 0;
        p0 = seqx[0] + pp[1].spc;
        p1 = seqx[1] + pp[0].spc;
        n0 = pp[1].spc + 1;
        n1 = pp[0].spc + 1;

        nm = 0;
        while ( *p0 && *p1 ) {
                if (siz0) {
                        p1++;
                        n1++;
                        siz0--;
                }
                else if (siz1) {
                        p0++;
                        n0++;
                        siz1--;
                }
                else {
                        if (xbm[*p0-'A']&xbm[*p1-'A'])
                                nm++;
                        if (n0++ == pp[0].x[i0])
                                siz0 = pp[0].n[i0++];
                        if (n1++ == pp[1].x[i1])
                                siz1 = pp[1].n[i1++];
                        p0++;
                        p1++;
                }
        }

        /* pct homology:
         * if penalizing endgaps, base is the shorter seq
         * else, knock off overhangs and take shorter core
         */
        if (endgaps)
                lx = (len0 < len1)? len0 : len1;
        else
                lx = (lx < ly)? lx : ly;
        pct = 100.*(double)nm/(double)lx;
        fprintf(fx, "\n");
        fprintf(fx, " < %d match%s in an overlap of %d: %.2f percent similarity\n",
                nm, (nm == 1)? "" : "es", lx, pct);
```

Table 1 (cont')

```
        fprintf(fx, "<gaps in first sequence: %d", gapx);                    ...getmat
        if (gapx) {
                (void) sprintf(outx, " (%d %s%s)",
                        ngapx, (dna)? "base":"residue", (ngapx == 1)? "":"s");
                fprintf(fx,"%s", outx);

        fprintf(fx, ", gaps in second sequence: %d", gapy);
        if (gapy) {
                (void) sprintf(outx, " (%d %s%s)",
                        ngapy, (dna)? "base":"residue", (ngapy == 1)? "":"s");
                fprintf(fx,"%s", outx);
        }
        if (dna)
                fprintf(fx,
                "\n<score: %d (match = %d, mismatch = %d, gap penalty = %d + %d per base)\n",
                smax, DMAT, DMIS, DINS0, DINS1);
        else
                fprintf(fx,
                "\n<score: %d (Dayhoff PAM 250 matrix, gap penalty = %d + %d per residue)\n",
                smax, PINS0, PINS1);
        if (endgaps)
                fprintf(fx,
                "<endgaps penalized. left endgap: %d %s%s, right endgap: %d %s%s\n",
                firstgap, (dna)? "base" : "residue", (firstgap == 1)? "" : "s",
                lastgap, (dna)? "base" : "residue", (lastgap == 1)? "" : "s");
        else
                fprintf(fx, "<endgaps not penalized\n");
}

static          nm;             /* matches in core -- for checking */
static          lmax;           /* lengths of stripped file names */
static          ij[2];          /* jmp index for a path */
static          nc[2];          /* number at start of current line */
static          ni[2];          /* current elem number -- for gapping */
static          siz[2];
static char     *ps[2];         /* ptr to current element */
static char     *po[2];         /* ptr to next output char slot */
static char     out[2][P_LINE]; /* output line */
static char     star[P_LINE];   /* set by stars() */

/*
 * print alignment of described in struct path pp[]
 */
static
pr_align()                                                          pr_align
{
        int             nn;     /* char count */
        int             more;
        register        i;

        for (i = 0, lmax = 0; i < 2; i++) {
                nn = stripname(namex[i]);
                if (nn > lmax)
                        lmax = nn;

                nc[i] = 1;
                ni[i] = 1;
                siz[i] = ij[i] = 0;
                ps[i] = seqx[i];
                po[i] = out[i];
        }
```

Page 3 of nwprint.c